Robust Throughput Boosting for Low Latency Dynamic Partial Reconfiguration

Nannarelli, Alberto; Re, M.; Cardarilli, Gian Carlo; Nunzio, L. Di ; Brunella, M. Spaziani ; Fazzolari, R. ; Carbonari, F.

Published in:
Proceedins of the 30th IEEE International System-on-Chip Conference

Publication date:
2017

Document Version
Peer reviewed version

Robust Throughput Boosting for Low Latency Dynamic Partial Reconfiguration

A. Nannarelli[(1)], M. Re, G. C. Cardarilli, L. Di Nunzio, M. Spaziani Brunella, R. Fazzolari and F. Carbonari
Department of Electronic Engineering, University of Rome "Tor Vergata", Rome, Italy
[(1)] DTU Compute, Technical University, Kongens Lyngby, Denmark

***Abstract*—Reducing the configuration time of portions of an FPGA at run time is crucial in contemporary FPGA-based accelerators. In this work, we propose a method to increase the throughput for FPGA dynamic partial reconfiguration by using standard IP blocks. The throughput is increased by over-clocking the configuration bitstream circuitry beyond the limits stated in the specifications of these standard blocks. The experimental results show that the most power efficient implementation can reach a throughput of about 780 MB/s, corresponding to a configuration latency of about 670 micro-seconds for bitstreams of 1.2 MB. We also investigate alternatives to boost the reconfiguration throughput and sketch a methodology to achieve the most power efficient implementation of FPGA-based accelerators.**

I. Introduction

Power efficiency is the key feature of contemporary computer systems: both for high-end and warehouse-scale computers, and for portable computers and embedded systems (Internet-of-Things, or IoTs).

A proven way to increase power efficiency is the use of hardware accelerators, which can perform the computation of specific applications rapidly and at lower energy [1]. Accelerators, such as GPUs, are employed in super-computers [2] and servers running heavy computations; e.g., machine learning [3], financial applications [4], etc..

In the "dark silicon" era [5], since the whole chip (die) cannot be powered simultaneously at the maximum performance, a viable solution is to have heterogeneous cores crafted for specific applications (Application Specific Processors, ASPs): a web server, a crypto engine, a decimal processor, etc..

However, because of the unlimited number of applications, especially in IoTs, it is nearly impossible to pack all the necessary ASPs in a specific System-on-Chip (SoC), since the SoC must be produced in high volumes to justify the production costs.

In this context, the implementation of ASPs in a reconfigurable fabric, such as FPGAs, can bridge the gap between acceleration for any computation-intensive application and low cost for low volumes. Moreover, since FPGAs can be re-configured on-the-fly (partial dynamic reconfiguration, or PDR), the same physical piece of silicon can be used to implement several ASPs, configured on demand.

For these reasons, the two major FPGA players, Xilinx and Intel (formerly Altera), offer SoC or SiP (System in a Package) solutions in which FPGA fabric to implement custom processors is placed next to cores [6], [7].

To efficiently use these reconfigurable accelerators on computing workloads, it is important to keep the configuration time (latency) as short as possible. With low reconfiguration latency, we can seamlessly change the hardware (ASP), similarly to what happens with dynamically loaded software routines, with negligible performance overheads.

In this work, we focus on the FPGA dynamic partial reconfiguration. Our approach, based on previous work [8], [9], resorts to over-clocking the hardware blocks necessary for the PDR. We present an analysis of the PDR system and experimental results of actual runs under different conditions.

Since over-clocking has an impact on the system's robustness, we test our design under different operating temperatures to see when the over-clocked implementation fails. The latter aspect is very important for industrial IoT computers working in harsh environments, such as factories.

The main contributions of this paper are:

- the design of an architecture for PDR to reduce the configuration latency based on over-clocking.
- Moreover, we measure the actual power dissipation in the FPGA chip at different die temperature to check the system's robustness and to determine which over-clocking rate gives the best trade-off throughput vs. energy (power efficiency).
- We also compare our method with recent related work using ad-hoc hardware blocks to see how distant is our approach based on standard blocks.
- Finally, we propose the design of an improved PDR system, based on a custom implementation of the memory interface to boost the reduce the reconfiguration latency even more.

The results of our experiments on power efficiency for the PDR system can be easily extended to any FPGA-placed hardware block when multiple clocking rates are supported.

II. Implementation Platform and Framework

The platform selected to implement the design is the Zed-Board hosting the Z-7020 device, of the *Xilinx Zynq-7000* SoC family [6]. The *Xilinx Zynq* comprises two different parts: the Processing System (PS) – a dual-core ARM Cortex-A9 with two NEON Media Processing Engines for SIMD operations – and the Programmable Logic (PL) – an FPGA based on the Xilinx Artix-7 technology.

The interconnection between the PS and the PL is provided by the AMBA AXI bus. The *AXI4-Stream* is the protocol

used to transfer large data sets; e.g., via direct memory access (DMA).

The PL part can be programmed, or configured, *statically* by loading the bitstream in the configuration memory before starting using the FPGA, or *dynamically* (partial reconfiguration) by loading bitstreams only to subsets of the configuration memory corresponding to selected areas of the FPGA.

The static configuration allows the maximum flexibility in terms of architecture of the processor implemented and can stretch to the maximum usable area of the PL.

The dynamic partial configuration, can be done on-the-fly on one portion of the PL, while other parts of the PL, keep running.

The configuration of the FPGA can be done by the PS through the Processor Configuration Access Port (PCAP), or by the Internal Configuration Access Port (ICAP), a hardware block that can read and write the FPGA configuration memory.

The ICAP allows to configure the FPGA directly from the PL. For this reason, full bitstreams (static configurations) cannot be loaded by the ICAP because the ICAP itself is part of the PL.

The architecture of our acceleration framework is designed taking into account several aspects, including:

- dynamic reconfiguration of reserved areas;
- clock rate adaptable to the specific ASP timing constraint;
- amount of data transferred from/to memory.

Fig. 1 shows the architecture of the accelerator with four reconfigurable portions (RP 1–4). In the figure, we identify the "static part" and the "dynamic part" of the PL.

The interface toward the PS side is through four high-performance ports (HP 0-3) which connect the PL to the ARM CPUs and the DRAM controller, and through the AXI ACP port which connects directly to the cache. In addition, the interface PS-PL provides four programmable clocks.

The static part is loaded statically by the PCAP and does not change on-the-fly. It consists of five *AXI4-Stream* blocks, the ICAP, and a clock management unit.

Since port ACP is not connected to the DRAM controller, the ASP connected to this port, cannot transfer large chunks of data. The maximum size is set by the cache size (512 KB).

Each of the four RPs in the dynamic part is dynamically reconfigured by the ICAP. Each RP can be connected to the PS through the 32-bit AXI GP ports using the *AXI4-Lite* bus.

Interrupts are used to signal change of status (end of configuration, data ready, etc.) in the RP areas to the PS.

Moreover, each RP can be clocked at a specific frequency thanks to the *"Clock Manager"* (Fig. 1), allowing maximum flexibility and IP-block reuse.

III. Architecture for ICAP Partial Configuration

The starting point of this work is the architecture for the ICAP-based partial reconfiguration presented in [9] and derived from [8]. As proposed in [8], by connecting an *AXI4-Stream* interface to the ICAP and transferring the bitstream via DMA, we obtain a transfer rate close to the theoretical limit of 400 MB/s. Then, we increase the clock rate (over-clocking)

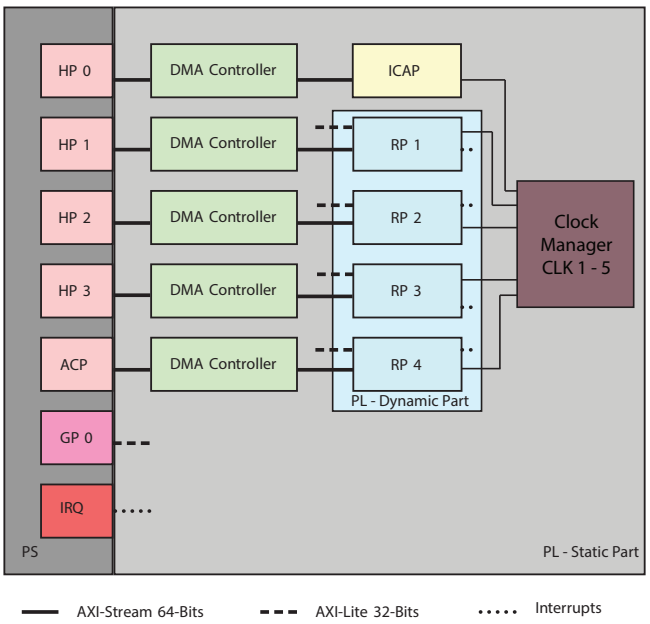

Fig. 1. HLL hardware architecture (not in scale).

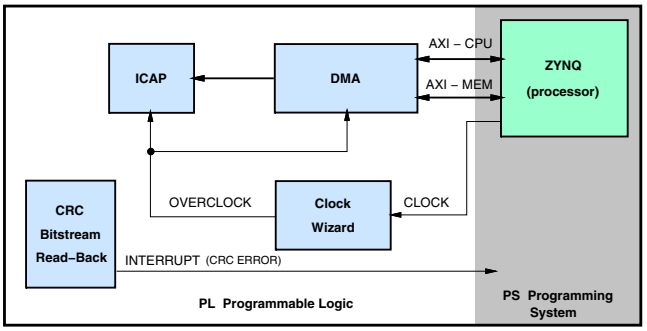

Fig. 2. Architecture for ICAP-based partial configuration.

of the hardware blocks controlling the partial configuration beyond the limits they are designed for, and we measure the impact on the throughput.

We use a Cyclic Redundancy Check (CRC) block, which reads back the configuration bitstream and checks the CRC of the configuration memory content, to determine whether the reconfiguration was successful.

The hardware architecture of our ICAP-based partial reconfiguration is detailed in Fig. 2. We briefly describe the relevant blocks.

The communication between the PS and the PL is handled by two AXI interconnects: one for the ARM cores and another to connect to the DRAM memory.

The **Clock Wizard** is a Xilinx IP core to adapt the circuit clock to the user's requirements. We use the Clock Wizard to set the clock frequency of our design under test. For testing purposes, we select the over-clocking frequency by the 8 switches on the ZedBoard. Clearly, for an actual system, this frequency will be set by a software command.

The over-clock signal operates both the **DMA** (Direct Memory Access) and **ICAP** (Internal Configuration Access

Fig. 3. OLED display: system status.

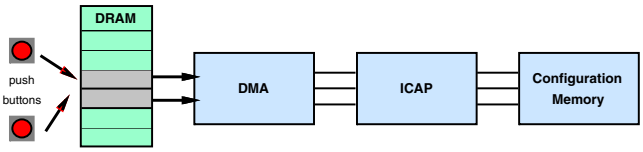

Fig. 4. Testing of the dynamic partial reconfiguration.

Port).

The ICAP block, created in previous work [9], reads and writes the FPGA configuration memory.

The **CRC Bitstream Read-Back** block reads back continuously in the background the whole bitstream to check the CRC of the configuration memory content. If a CRC error is detected an interrupt is asserted.

The hardware blocks are completed by a temperature sensing block and an OLED display controller, not depicted in Fig. 2.

The OLED panel (Fig. 3) displays the current over-clocking frequency and chip temperature, the result of CRC testing, and the partial bitstream transfer time.

The whole hardware in the PL is controlled by a C program running in the PS.

IV. Experimental Results

The testing flow of our over-clocked partial reconfiguration scheme is shown in Fig. 4.

The application software used to test the system is loaded on an SD memory card. The Zedboard is booted from the SD card. The memory card also contains two bitstreams, about 1.2 MB in size, to partially reconfigure a selected area of the FPGA (RP 1–4 in Fig. 1).

Table I shows the throughput achieved in transferring the bitstreams to the configuration memory by the over-clocked ICAP+DMA.

The throughput is computed off-line by dividing the bitstream size by the configuration latency, determined by the C-timer and displayed on the OLED display.

We use the Zedboard's switches to set the over-clocking frequency. Moreover, we use two push-buttons to start the ICAP operations and load one of the two bitstreams. The testing results are displayed on the OLED screen. This setup can be used to test any over-clocked block.

ICAP Frequency $[MHz]$	Config. Latency $[\mu s]$	Throughput $[MB/s]$	CRC
100	1325.60	399.06	valid
140	947.40	558.12	valid
180	737.50	716.96	valid
200	676.30	781.84	valid
240	671.90	786.96	valid
280	669.20	790.14	valid
310	N/A no interrupt	N/A	valid
320	N/A no interrupt	N/A	not valid
360	N/A no interrupt	N/A	not valid

TABLE I
THROUGHPUT VS. FREQUENCY WHEN OVER-CLOCKING.

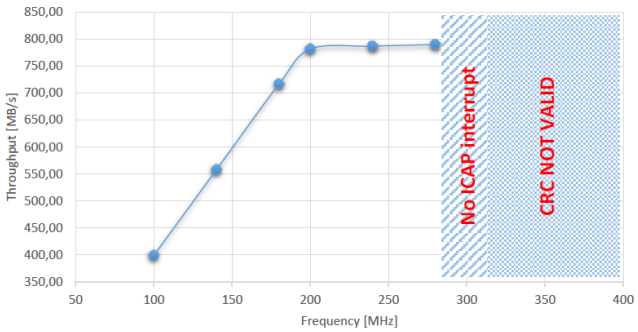

Fig. 5. Throughput vs. frequency.

We tested several over-clocking frequencies, starting from 100 MHz which is the nominal frequency the DMA and ICAP blocks are designed for.

The system stopped working when over-clocked at 310 MHz, where the CRC block never asserted the interrupt. For higher clock rates, also the CRC value resulted in error (bitstream not correctly loaded).

By plotting the results of our test in a throughput-frequency plane (Fig. 5), we can better see that the throughput increases linearly until about 200 MHz when the curve flattens.

This saturation is due to the AXI DMA, which is guaranteed to work up to 150 MHz in Xilinx's Application Notes. We found that, for our system, this limit can be raised up to 310 MHz, although, above 200 MHz, the performance improvements are marginal.

In conclusion, we measured that the maximum throughput for an ICAP-based partial bitstream transfer is 790 MB/s at 280 MHz. However, the knee-bend performance is at 200 MHz with a throughput of 782 MB/s. At higher clock frequencies, the bitstream transfer fails and the CRC block correctly detects an error.

A. Temperature Stress

To check how robust to temperature variations our over-clocking scheme is, we heated up the Zynq chip with a heat gun by concentrating the heat on the Zynq's heat sink and by keeping the rest of the ZedBoard at room temperature. The die temperature is measured by the built-in temperature sensor and read on the OLED display.

We performed again the tests of Table I up to 310 MHz starting from a die temperature of 40^{o}C until 100^{o}C, at steps of

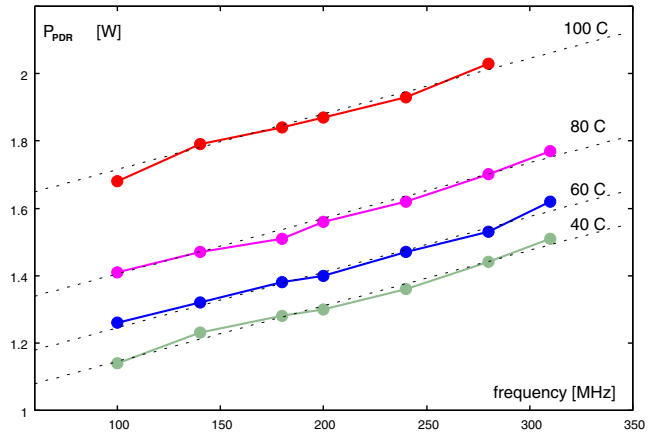

Fig. 6. Power dissipation in the Zynq SoC at different frequencies and temperatures.

10^{o}C. All the tests succeeded except the test done at 310 MHz and 100^{o}C which failed. This proves that the over-clocking is robust enough to withstand the temperature variations typical of harsh computing environments.

B. Power Efficiency

The ZedBoard hosting the Zynq SoC is equipped with current-sense pin-headers to measure the power dissipation of the whole board. We measured an average power dissipation of $P_0 = 2.2\ W$ (at temperature 40^{o}C) when the Zynq SoC is idle (no processing in the PS) and the PL is not programmed. Then, we activated the Zynq and run the reconfiguration experiments at the different clock frequencies (100–310 MHz) and temperature steps of 10^{o}C from 40^{o}C to 100^{o}C. We consider the power consumption in the Zynq only

$$P_{PDR} = P_f^T - P_0 \qquad [W]$$

where P_f^T is the actual reading at frequency f and temperature T, and P_0 is our reference power dissipation when no computation is performed in the Zynq (at 40^{o}C).

The measured data points are plotted in Fig. 6. For clarity, we only plot the values obtained for temperature steps of 20^{o}C.

By considering the power-frequency relationship, Fig. 6 shows that the dynamic power dissipation increases linearly with frequency and that the slope is constant at the different temperatures. In other words, the dynamic power dissipation is not affected by the die temperature in our experiments.

As for the static power dissipation, Fig. 6 confirms the more than linear increase of power with temperature.

By combining the data on the throughput and the power dissipation, we can evaluate the power efficiency of our PDR by ICAP defined as *"performance-per-watt" (PpW)*:

$$\text{PpW} = \frac{\text{throughput}}{P_{PDR}} \qquad \left[\frac{\text{MBytes}}{s \cdot W = J}\right]$$

Table II shows that since the throughput plateaus at 200 MHz, but P_{PDR} keeps increasing with frequency, the most power efficient implementation is about 600 MB/J at 200 MHz.

Frequency $[MHz]$	P_{PDR} $[W]$	Throughput $[MB/s]$	Power Eff. $[MB/J]$
100	1.14	399.06	351
140	1.23	558.12	453
180	1.28	716.96	560
200	1.30	781.84	**599**
240	1.36	786.96	577
280	1.44	790.14	550

TABLE II
POWER EFFICIENCY FOR OVER-CLOCKING AT 40^{o}C.

V. Related Work

We now compare our over-clocked dynamic partial reconfiguration environment with other relevant work in the area. The results of the comparison are reported in Table III.

1) VF-2012: The authors of [8], presented in [10] an over-clocked ICAP controller to maximize throughput. The throughput of 400 MB/s at the nominal clock of 100 MHz scales nicely to 838.55 MB/s at 210 MHz. Above this frequency, the reconfiguration fails and above 300 MHz, initiating a reconfiguration, freezes the whole FPGA. No CRC is implemented in [10].

2) HP-2011: In [11], the authors proposed to use a multi-port memory controller to interface the ICAP. Furthermore, they also introduce over-clocking with active feedback to ensure that the device voltages and temperatures are within nominal values. The maximum throughput achieved (Xilinx Virtex-5) is about 420 MB/s at 133 MHz.

3) HKT-2011: In [12], the authors present the design of an enhanced hard macro for the ICAP. For different system clocks, the authors use the ICAP hard macro with different over-clocking methods and achieve a maximum throughput of 2200 MB/s by clocking the ICAP at 550 MHz. The system, implemented on a Xilinx Virtex-5, does not include a processor and the configuration bitstreams (up to 50 KB) are buffered in a FIFO.

Design	Platform	ICAP frequency $[MHz]$	Throughput $[MB/s]$
VF-2012	Virtex-6	210	839
HP-2011	Virtex-5	133	419
HKT-2011	Virtex-5	550	2200
This work	Zynq-7000	280	790

TABLE III
COMPARISON WITH RELATED WORK.

By comparing the data in Table III, our proposed over-clocked ICAP has the same performance of VF-2012 at nominal 100 MHz. While the throughput scales linearly up to 210 MHz in VF-2012, our system saturates above 200 MHz when the DMA cannot keep the bus full in every cycle, However, our system performs a CRC to automatically detect errors. In addition, we perform a power dissipation and temperature analysis to provide more insight in the PDR.

With respect to the implementation HP-2011, our over-clocked ICAP offers a higher throughput. However, HP-2011 is implemented in an older technology and the gap might be narrower if scaled to the same technology.

The implementation HKT-2011 shows the highest throughput, but this is obtained for small sized bitstreams. Since what really matters for on-the-fly PDR is the reconfiguration latency, it is very hard to assess if the 2200 MB/s throughput can be sustained through a DMA necessary to transfer bitstreams of about 1.4 MB.

VI. Proposed Partial Reconfiguration Environment

The system that provides the bitstream transfer from the DRAM system memory to the PL through ICAP is not the best in terms of throughput performance. The bottleneck is located within the link Memory Port → AXI Interconnect → AXI DMA.

Here we propose a new dedicated system, based on [12], that uses a different interconnection, thus obtaining the maximum speed from the ICAP transfer.

Our target is to redesign the ICAP transfer as sketched in Fig. 7. The functionality of the architecture is explained next.

The partial bitstream is pre-loaded in the **SRAM** for the reconfiguration to take place at the highest possible throughput. The SRAM memory can store one partial bitstream a time, thus only one hardware IP block.

The **Memory Controller** manages the write and read tasks of the SRAM. Specifically, it generates the read/write addresses.

The **PR** (Partial Reconfiguration) **Controller** is an arbiter between the SRAM and the ICAP: it monitors the reconfiguration timing and the ICAP interrupts.

The **Bitstream Decompressor** decompresses the bitstream.

The **PS Scheduler** is linked with the control blocks to manage the SRAM write procedure (from the DRAM). It manages all the partial bitstreams that are needed by the whole architecture, and it pre-loads the next on the SRAM, whilst, for example, the current partially configurable hardware accelerator is performing its task.

By exploring state-of-the-art memory devices, we found a SRAM from Cypress (CY7C2263KV18) that meets our throughput requirements: Double Date Rate (DDR) interfaces on both read and write ports at 550 MHz. The read access time is 0.45 ns.

Based on the maximum clock frequency and the required data bus (36 bit), the maximum throughput is

$$\text{throughput} = 550\ MHz \cdot \frac{36\text{ bit}}{2} = 1237.5\ \ [MB/s].$$

This theoretical throughput is almost double the one measured by the current system in Sec. IV.

VII. Conclusions

In this work we presented a method to reduce the reconfiguration latency for FPGA-based accelerators by boosting the throughput for dynamic partial reconfiguration. The throughput is increased by over-clocking the DMA and ICAP beyond the limits stated in the specifications of these standard IP blocks.

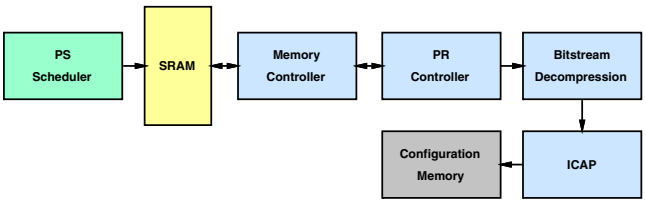

Fig. 7. Proposed partial reconfiguration via ICAP.

The experimental results show that, from the nominal throughput of about 400 MB/s, we can reach a maximum throughput of 790 MB/s by over-clocking to 280 MHz. However, by considering the power dissipation as well, the most power efficient implementation of the dynamic partial reconfiguration is at a throughput of 782 MB/s at 200 MHz, that is a transfer of about 600 MB per Joule, and about 670 μs for 1.2 MB bitstreams typical for our ASPs.

Moreover, inspired by related work, we investigated alternatives to boost the reconfiguration throughput by exploring the design of SRAM-based custom IP blocks and interfaces to deliver an estimated 1250 MB/s throughput.

The power dissipation and temperature analysis, presented in Sec. IV for the over-clocked ICAP, can be extended to any IP block implemented in the FPGA to determine its best trade-off throughput vs. energy, and design the most power efficient accelerator for the specific application and platform.

References

[1] S. Patel and W.-M. W. Hwu, "Accelerator Architectures," *IEEE Micro magazine*, vol. 28, pp. 4–12, July/Aug. 2008.

[2] Top 500 Supercomputer Sites. Top 500 List. [Online]. Available: http://www.top500.org/lists/

[3] E. Nurvitadhi *et al.*, "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?" in *Proc. of ACM FPGA'17*, Feb. 2017.

[4] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "What is Happening to Power, Performance, and Software?" *IEEE Micro*, vol. 32, no. 3, pp. 110–121, May-June 2012.

[5] M. B. Taylor, "A Landscape of the New Dark Silicon Design Regime," *IEEE Micro*, vol. 33, no. 5, pp. 8–19, Sep.-Oct. 2013.

[6] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC*, 1st ed. Strathclyde Academic Media, 2014.

[7] J. Burt, "Intel Begins Shipping Xeon Chips With FPGA Accelerators," *eWeek*, Apr. 2016. [Online]. Available: http://www.eweek.com/servers/intel-begins-shipping-xeon-chips-with-fpga-accelerators.html

[8] K. Vipin and S. A. Fahmy, "ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq," *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, Sep. 2014.

[9] A. Lomuscio, G. Cardarilli, A. Nannarelli, and M. Re, "A Hardware Framework for on-Chip FPGA Acceleration," in *to appear in Proc. Int.l Symposium on Integrated Circuits (ISIC 2016)*, Dec. 2016.

[10] K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA Partial Reconfiguration," in *2012 International Conference on Field-Programmable Technology (FPT)*, Dec. 2012, pp. 61–66.

[11] J. C. Hoffman and M. S. Pattichis, "A High-Speed Dynamic Partial Reconfiguration Controller Using Direct Memory Access Through a Multiport Memory Controller and Overclocking with Active Feedback," *International Journal of Reconfigurable Computing*, vol. 2011, no. ID 439072, p. 10, 2011.

[12] S. G. Hansen, D. Koch, and J. Torresen, "High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, May 2011, pp. 174–180.